FIG. 1A

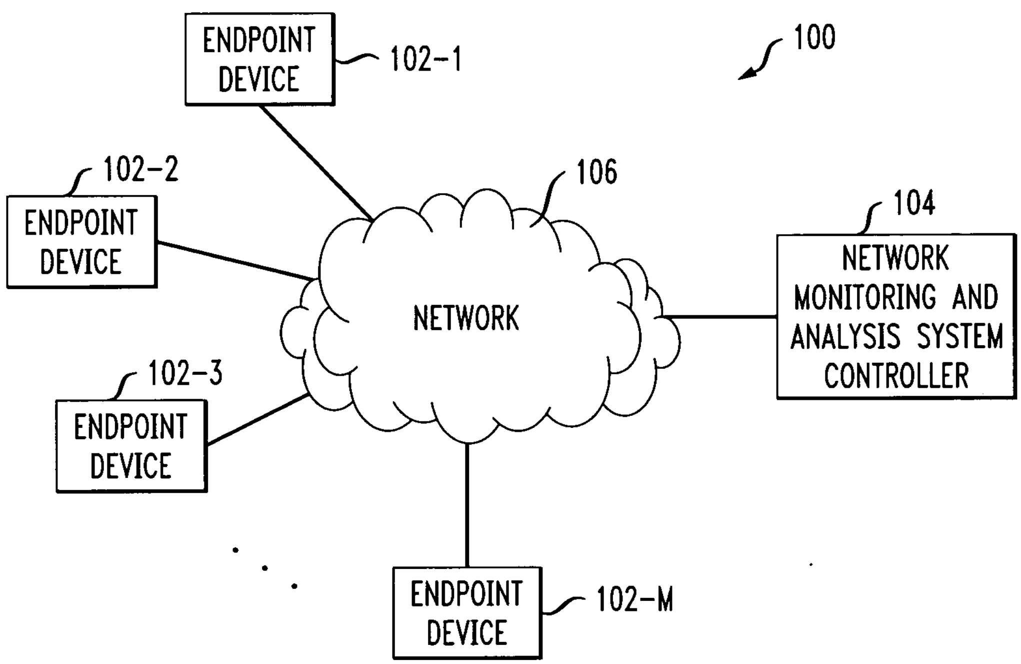
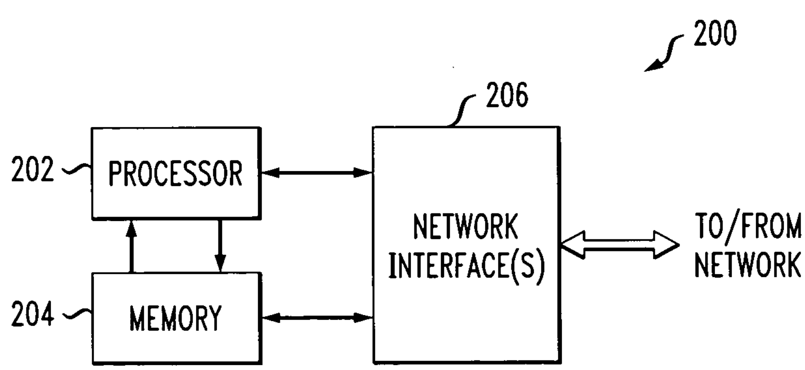

100
ENDPOINT DEVICE
102-1
102-2
ENDPOINT DEVICE
106
104
NETWORK MONITORING AND ANALYSIS SYSTEM CONTROLLER
NETWORK
102-3
ENDPOINT DEVICE
ENDPOINT DEVICE
102-M

FIG. 1B


200
206
202
PROCESSOR
NETWORK INTERFACE(S)
TO/FROM NETWORK
204
MEMORY

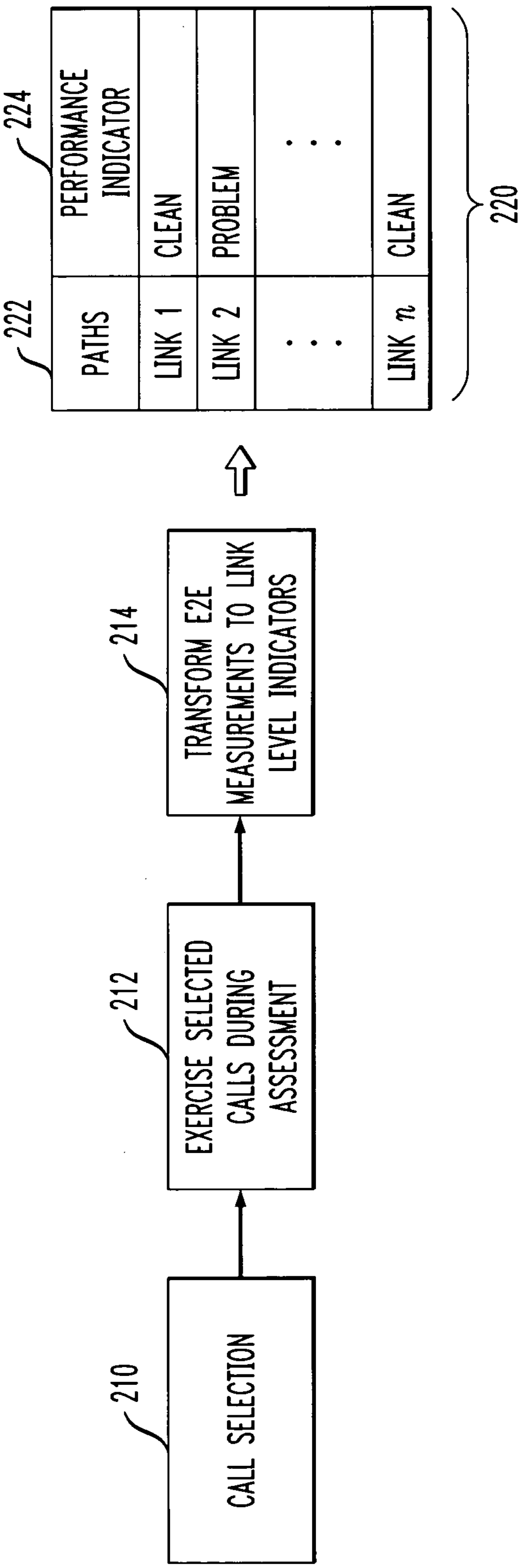

210
CALL SELECTION
212
EXERCISE SELECTED CALLS DURING ASSESSMENT
214
TRANSFORM E2E MEASUREMENTS TO LINK LEVEL INDICATORS
222
224
PATHS
PERFORMANCE INDICATOR
LINK 1
CLEAN
LINK 2
PROBLEM
. . .
. . .
LINK n
CLEAN
220

FIG. 2A

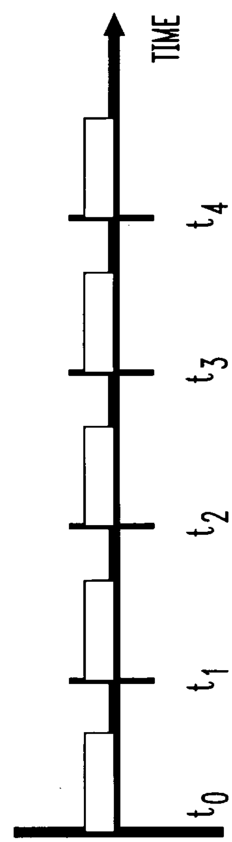

TIME
t_0
t_1
t_2
t_3
t_4

FIG. 2B

FIG. 3

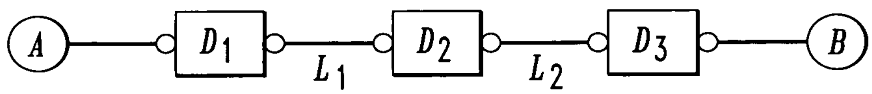


FIG. 4

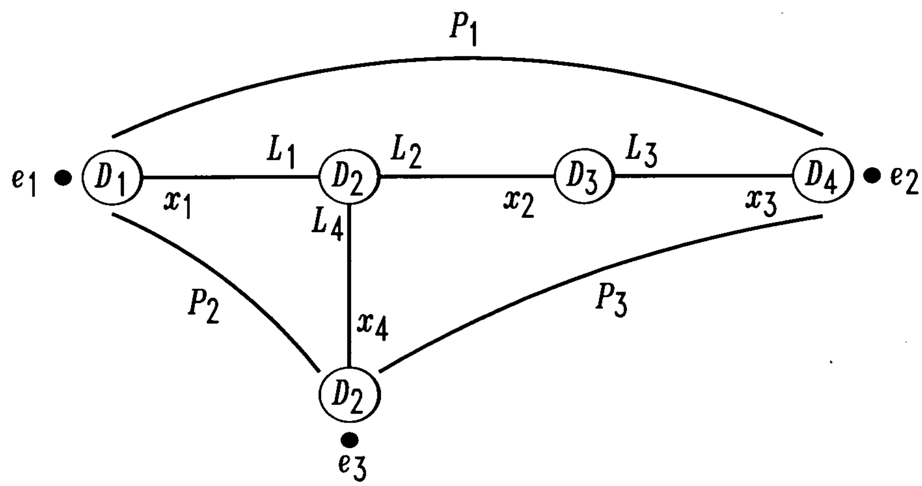


FIG. 5

	L_1	L_2	L_3	L_4
P_1	1	1	1	0
P_2	1	0	0	1

FLOW MATRIX 1

	L_1	L_2	L_3	L_4
P_1	1	1	1	0
P_2	1	0	0	1
P_3	0	1	1	1

FLOW MATRIX 2

FIG. 6

EQUATIONS WITH FLOW MATRIX 1

$$\begin{aligned} x_1 + x_2 + x_3 &= y_1 \\ x_1 + x_4 &= y_2 \end{aligned} \qquad \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

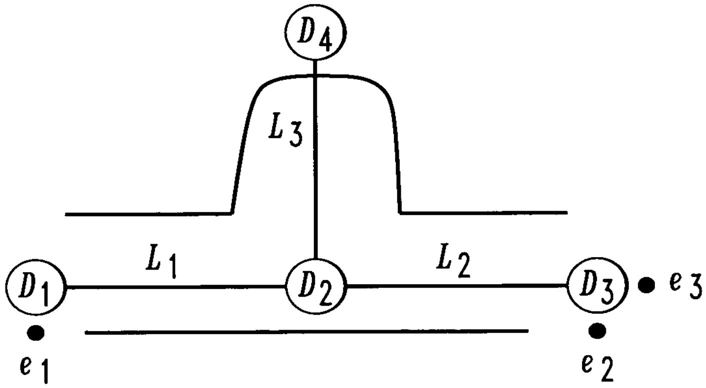
EQUATIONS WITH FLOW MATRIX 2

$$\begin{aligned} x_1 + x_2 + x_3 &= y_1 \\ x_1 + x_4 &= y_2 \\ x_2 + x_3 + x_4 &= y_3 \end{aligned} \qquad \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

FIG. 7

Generate_Pipes($G = (D, L)$): **Network Topology Graph**, E: **Set of Leaves**)
I: Set of pipes in G wrt E
$I \leftarrow \emptyset$
Compute P for G wrt E
Let M be the complete flow matrix for G and P
//Group links with the same column vector into disjoint sets
Let k be the number of distinct column vectors in M
Form a set **S** $= \{S_0, S_1, \ldots, S_k\}$ where:
  each S_i, $0 < i \leq k$ contains links in L with the i^{th} distinct column vector in M
//Ensure that links in each element of S form a path in G
for i=1 to |**S**|
  if links in S_i are consecutive and form a path
    then merge S_i into path p, $I \leftarrow I \cup \{p\}$ *//add the path formed by the links in S_i as a pipe*
    else $I \leftarrow I \cup S_i$ *//add each link as a pipe by itself*
return I

FIG. 8



$$\left[\begin{array}{l|ccc} & L_1 & L_2 & L_3 \\ \hline e_1 - e_2 & 1 & 1 & 0 \\ e_1 - e_3 & 1 & 1 & 2 \\ e_2 - e_3 & 0 & 0 & 0 \end{array}\right]$$

FIG. 9

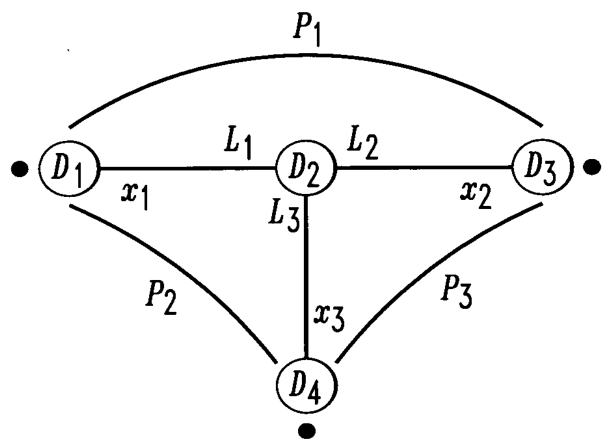


FIG. 10

FLOW MATRIX 1

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

FLOW MATRIX 2

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

FIG. 11

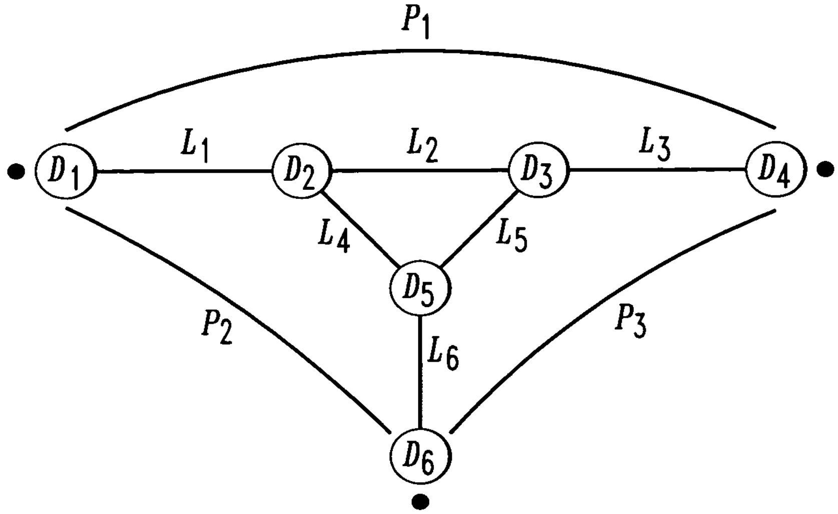


FIG. 12

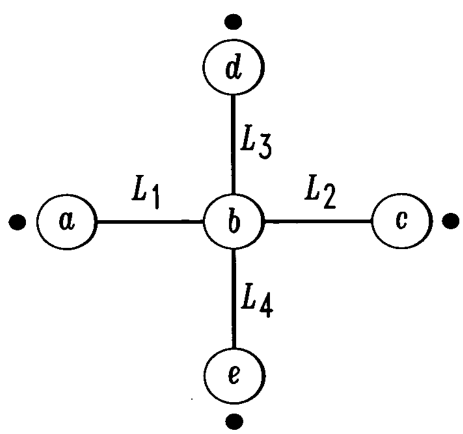


FIG. 13

	L_1	L_2	L_3	L_4
$L_1.L_4$	1	0	0	1
$L_4.L_2$	0	1	0	1
$L_2.L_3$	0	1	1	0
$L_3.L_1$	1	0	1	0

$\{L_1.L_4,\ L_4.L_2,\ L_2.L_3,\ L_3.L_1\}$

	L_1	L_2	L_3	L_4
$L_1.L_2$	1	1	0	0
$L_1.L_3$	1	0	1	0
$L_1.L_4$	1	0	0	1
$L_2.L_3$	0	1	1	0

$\{L_1,\ L_2,\ L_3,\ L_4\}$

FIG. 14

Select_Matrix($G' = (D', I)$: Reduced Network Topology Graph, E: Set of Leaves)

```
W: Set of worms in G' wrt E, W ← Ø
R: Set of paths, R ← Ø
Compute P' for G' wrt E
open ← P'
while open ≠ Ø
    select p from open
    for each pipe c_i on p = c_1.c_2. ...c_length(p)
      if ∃S ⊂ open such that S makes c_i estimable                    ← B1
          Compute S' which has the original value of each path in S
          R ← R ∪ S'
          W ← W ∪ {c_i}
              update open and W such that ∀p' ∈ open
                p' does not contain any estimable path in W       //c_i is removed from paths in open
      else
          c_{i+1} ← c_i.c_{i+1}
    open ← open \ {p}
return W, R
```

FIG. 15

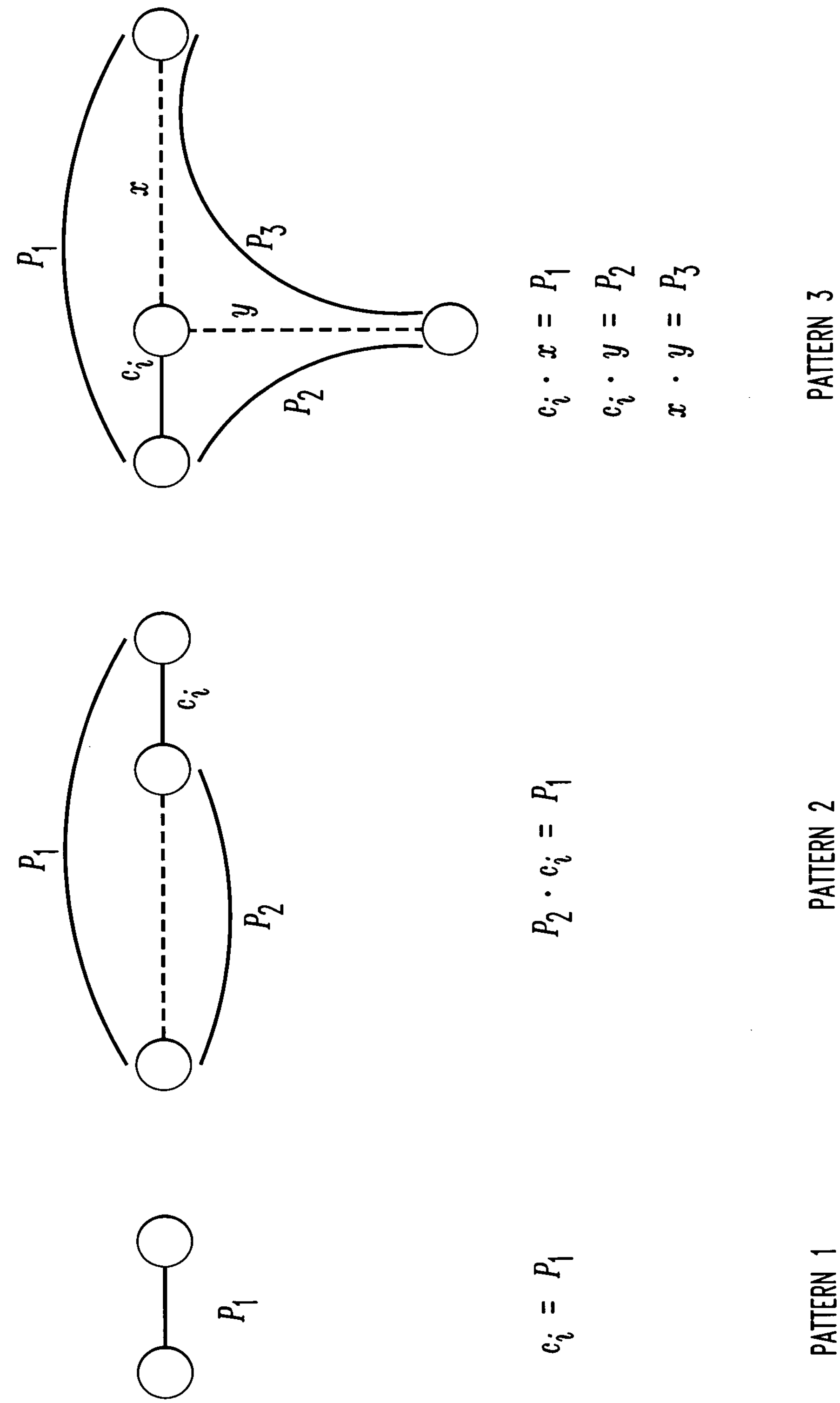

P_1
$c_i = P_1$
PATTERN 1
P_1
P_2
c_i
$P_2 \cdot c_i = P_1$
PATTERN 2
P_1
x
P_3
y
c_i
P_2
$c_i \cdot x = P_1$
$c_i \cdot y = P_2$
$x \cdot y = P_3$
PATTERN 3

FIG. 16

```
Compute_EstPaths(G' = (D', I):Reduced Network Topology Graph, E: Set of Leaves, P'_{t_i} : End-to-end paths at time t_i)

M: A Minimal set of estimable paths for G' wrt E, W ← Ø
open ← P'_{t_i}
while open ≠ Ø
    while open not converged
        select p from open
        for each pipe c_i on p = c_1.c_2. ...c_{length(p)}
          if ∃ S ⊂ open such that S makes c_i estimable
                M ← M ∪ {c_i}
                update open and M such that ∀p' ∈ open
                  p' does not contain any estimable path in M and        //c_i is removed from paths in open
                  open ← open \ {p}
        else
            abort processing of p
    if open ≠ Ø
        select shortest p in open
        open ← open \ {p}
        M ← M ∪ {p}
return M
```